REPORT DOCUMENTATION PAGE

DTIC ELECTE JUN 3 0 1988 D

AD-A196 246

H

DTIC FILE COPY

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-0655	
6a. NAME OF PERFORMING ORGANIZATION University of Maryland	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research	
6c. ADDRESS (City, State, and ZIP Code) College Park, Maryland 20742		7b. ADDRESS (City, State, and ZIP Code) Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR	8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-82-0078	
8c. ADDRESS (City, State, and ZIP Code) Bolling AFB DC 20332-6448		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO
61102F	2304	A3	

11. TITLE (Include Security Classification)
PARALLEL MATRIX COMPUTATIONS

12. PERSONAL AUTHOR(S)
G. W. Stewart, Dianne P. O'Leary

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL	FROM 1982 TO 1987	88-03-11	11

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This project concerns the design and analysis of algorithms to be run in a processor-rich environment. We focus primarily on algorithms that require no global control and that can be run on systems with only local connections among processors. We investigate the properties of these algorithms both theoretically and experimentally. The experimental work is done on the ZMOB, a working parallel computer operated by the Laboratory for Parallel Computation of the Computer Science Department at the University of Maryland.

To give our work direction, we have focused on two areas:

1. Dense problems from numerical linear algebra, and
2. The iterative and direct solution of sparse linear systems.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL John Thomas	22b. TELEPHONE (Include Area Code) (202)-767-5026 / 22c. OFFICE SYMBOL NM

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Technical Summary Report
AFOSR 82-0078
Parallel Matrix Computations
1982-1987

Supported by
Air Force Office of Scientific Research
Numerical Mathematics
Bolling Air Force Base, D.C. 20332

Research conducted at
Department of Computer Science
University of Maryland
College Park, MD 20742
(301) 454-2001

Principal Investigators
Professor G. W. Stewart (410-68-8197)
Assoc. Prof. Dianne P. O'Leary (359-46-2895)

1. INTRODUCTION

This project concerned the design and analysis of algorithms to be run in a processor-rich environment. We focused primarily on algorithms that require no global control and that can be run on systems with only local connections among processors. We investigated the properties of these algorithms both theoretically and experimentally. The experimental work was done on the ZMOB, a parallel computer operated by the Laboratory for Parallel Computation of the Computer Science Department at the University of Maryland, and on the BBN Butterfly computer as well.

The ZMOB consists of 128 processors which communicate by message passing over a communications network which provides a complete network of connections between processors. The start-up time for interprocessor communication, the per-word transmission overhead, and the floating point computation time are all of the same order of magnitude. The ZMOB appears to a user to be a completely connected network and thus can simulate various locally connected networks of processors. Thus we could investigate, in a realistic setting, the effects on our algorithms of various processor interconnections.

The BBN Butterfly is a 16 or 128 processor machine with a memory module locally assigned to each processor, globally accessible through a multilevel switch. It, too, can be used to simulate various processor interconnections for message-passing computers, and we also used it as a tool to understand algorithm performance on a locally-shared memory computer under construction at the University of Maryland.

Our activities may be divided into four categories: algorithms, software development, theoretical analysis, and experimental analysis.

To give our work direction, we focused on dense and sparse problems from numerical linear algebra. We discuss in this summary the research projects that we have pursued under this grant support.

2. Summary of Work

Our activities ranged from theoretical analysis to algorithmic design and software development. We summarize this work in the following sections. For details, consult the annotated list of references in Appendix A.

2.1 Data-Flow Algorithms and Domino

We based most of our work in this area on the notion of a *data-flow algorithm*. The computations in a data-flow algorithm are done by independent *computational nodes*, which cycle between requesting data from certain nodes, computing, and sending data to certain other nodes. More precisely, the nodes lie at the vertices of a directed graph whose arcs represent lines of communication. Each time a node sends data to another node, the data is placed in a queue on the arc between the two nodes. When a node has requested data from other nodes, it is blocked from further execution until the data it has requested arrives at the appropriate input queues. An algorithm organized in this manner is called a data-flow algorithm because the times at which nodes can compute is controlled by the flow of data between nodes.

Data-flow algorithms are well suited for implementation on networks of processors which communicate by message passing. Each node in a computational network is regarded as a process residing on a fixed member of a network of processors. We allow more than one node on a processor. Since many nodes will be performing essentially the same functions, we allow nodes which share a processor also to share pieces of reentrant code, which we call *node programs*. Each processor has a resident node communication and control system to receive and transmit messages from other processors and to awaken nodes when their data has arrived.

Data-flow algorithms have a number of advantages.

1. The approach eliminates the need for global synchronization.
2. Parallel matrix algorithms, including all algorithms for systolic arrays, have data-flow implementations.
3. Data-flow algorithms can be coded in a high-level sequential programming language, augmented by two communication primitives for sending and receiving data.
4. Data-flow computations can be supported by a very simple node communication and control system.
5. The approach allows the graceful handling of missized problems, since several nodes can be mapped onto one processor.
6. By mapping all nodes in a data-flow algorithm onto a single processor, one can debug parallel algorithms on an ordinary sequential processor.

Because of the conceptual convenience and practical utility of the data-flow approach, and because of the absence of any standard for writing transportable algorithms for parallel machines, we implemented these ideas in a node communication and control system called Domino. During 1985-86, we documented the system and provided examples of its use in a technical report. The code is available through Arpanet from Netlib at Argonne National Laboratories. Later, the system was implemented on the BBN Butterfly, McMob (a faster 16 processor version of Zmob), and on a single processor version of the Sequent. It is being used at various universities around the country, and also runs on the ZMOB, Vaxes under Unix or VMS, Sun workstations, and IBM PC's. The code has been used for numerical analysis and for neural network studies at Maryland and for real time algorithms and other applications at other universities. The system has been very valuable to us in our research, and was used in a course on parallel computation taught at Maryland. One of the students made enhancements to the system by incorporating a window system to use on the IBM-PC for debugging. We are working on a revision of the system to make it more efficient on shared memory systems.

2.2 Theoretical Developments

Work has been done in the design of parallel architectures and in the analysis of parallel algorithms.

Our work on the determinacy of our data-flow model for parallel computation led us to propose a modification of the design of systolic arrays in order to eliminate the need for global synchronization. Each cell in the array is augmented by a feedback circuit so that data is sent from one cell to another only when the receiver is ready to process it. We call such networks ***systaltic arrays***.

In collaboration with hardware experts Mark Weiser and Roger Pierson, we designed a processing module called the Maryland Crab, which can communicate with a limited number of neighbors directly through access to their memories. A prototype is currently under construction. These modules can be linked in various configurations to create parallel processing machines. They allow much faster message-passing than standard networks of processors, and avoid the complications of increasing depth of the switch that plague shared memory computers. Because of this we hope that architectures based on modules of this type can be extended to very large (>10,000) processor arrays.

This architecture work has motivated an extensive calculation to determine what arithmetic and communication speeds are needed to perform matrix computations with a large (>10,000) number of processers.

David C. Fisher completed a thesis partially supported by this grant which studies the complexity of various tasks in matrix computation, assuming that processors perform computations so fast that the communication delay in sending between physically distant processors is significant. Lower bounds on execution time were obtained, and optimal algorithms were derived for several problems.

The analysis of parallel numerical algorithms has to be understood in two senses. In the first place there are the conventional analyses that must be done on any numerical algorithm; rounding error analyses, proofs of convergence, and determination of rates of convergence are typical examples. In the course of developing algorithms we have done a number of these. Beyond these analyses there is the problem of determining how well a parallel implementation works. This is analogous to the computation of operations counts and other performance measurements for sequential algorithms. The main part of our theoretical work is devoted to the study of this problem. We have considered three issues: determinacy, assignment, and scheduling.

The determinacy issue arises from the fact that in the specification of a data-flow algorithm, there may be no unique order of execution for the nodes. Thus it was necessary to show that whatever the order, the computation produces essentially the same results.

The issues of assignment and scheduling are closely related. When a computational network is to be mapped onto a smaller network of processors, it may happen that there are several ways of assigning the nodes to processors. The question then arises of which way is best. Once several nodes are executing on a processor, an operating system such as Domino must schedule the nodes which are ready for execution according to some fixed strategy. Again the question arises of which scheduling strategy is best. The assignment and scheduling issues are related because an optimal scheduling strategy for one assignment may not be optimal for another.

We investigated these issues for a class of algorithms for matrix factorization, including implementations of the Cholesky algorithm, the LU decomposition, and the QR decomposition. We identified several good assignment and scheduling strategies for problems in which the number of matrix elements exceeds the number of processors, and computed upper and lower bounds on the execution times. This permits choice of a good algorithm for a particular machine, once the ratio of computation time to communication time is known.

2.3 Algorithm Design, Analysis, and Testing

The chief difficulty with the data-flow approach is that the behavior of the algorithms cannot be analyzed purely from the local viewpoint of the node programs. This is one reason for supplementing theory with experiment.

We devised a number of new parallel algorithms. For dense matrices we developed a parallel version of the QR algorithm for computing eigenvalues, motivated by the fast message passing capability of the Crab. We also ran a series of experiments to validate a theoretical model of the performance of parallel algorithms for QR factorization of matrices. For sparse matrices, we analyzed and reported on block conjugate gradient algorithms for solving linear systems.

John Conroy completed a doctoral thesis partially supported by this grant which studies efficient algorithms for solution of narrow and wide banded linear systems on parallel processors, and medium grained algorithms for nested dissection.

3. Summary

Our work resulted in a collection of parallel algorithms for matrix computations, a data-flow operating system to support experiments, a proposal for machine architecture, and theoretical investigation into complexity issues in parallel matrix computations.

Appendix
Accomplishments under Grant AFOSR 82-0078

I. Technical Reports

(1) G. W. Stewart, *Computing the CS Decomposition of a Partitioned Orthonormal Matrix*, TR-1159, May, 1982.

This paper describes an algorithm for simultaneously diagonalizing by orthogonal transformation the blocks of a partitioned matrix having orthonormal columns.

(2) G. W. Stewart *A Note on Complex Division*, TR-1206, August, 1982.

An algorithm (Smith, 1962) for computing the quotient of two complex numbers is modified to make it more robust in the presence of underflows.

(3) D. P. O'Leary, *Solving Sparse Matrix Problems on Parallel Computers*, TR-1234, December, 1982.

This paper has a dual character. The first part is a survey of some issues and ideas for sparse matrix computation on parallel processing machines. In the second part, some new results are presented concerning efficient parallel iterative algorithms for solving mesh problems which arise in network problems, image processing, and discretization of partial differential equations.

(4) G. W. Stewart, *A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix*, TR-1321, August, 1983.

This paper describes an iterative method for reducing a general matrix to upper triangular form by unitary similarity transformations. The method is similar to Jacobi's method for the symmetric eigenvalue problem in that it uses plane rotations to annihilate off-diagonal elements, and when the matrix is Hermitian it reduces to a variant of Jacobi's method. Although the method cannot compete with the QR algorithm in serial implementation, it admits of a parallel implementation in which a double sweep of the matrix can be done in time proportional to the order of the matrix.

(5) Dianne P. O'Leary and Robert E. White, *Multi-Splittings of Matrices and Parallel Solution of Linear Systems*, TR-1362, December, 1983.

We present two classes of matrix splittings and give applications to the parallel iterative solution of systems of linear equations. These splittings generalize regular splittings and P-regular splittings, resulting in algorithms which can be implemented efficiently on parallel computing systems. Convergence is established, rate of convergence is discussed, and numerical examples are given.

(6) D. P. O'Leary and G. W. Stewart, *Data-Flow Algorithms for Matrix Computations*, TR-1366, January, 1984.

In this work we develop some algorithms and tools for solving matrix problems on parallel processing computers. Operations are synchronized through data-flow alone, which makes global synchronization unnecessary and enables the algorithms to be implemented on machines with very simple operating systems and communications protocols. As examples, we present algorithms that form the main modules for solving Liaponuv matrix equations. We compare this approach to wavefront array processors and systolic arrays, and note its advantages in handling missized problems, in evaluating variations of algorithms or architectures, in moving algorithms from system to system, and in debugging parallel algorithms on sequential machines.

(7) G. W. Stewart, W. F. Stewart, D. F. McAlister, *A Two Stage Iteration for Solving Nearly Uncoupled Markov Chains*, TR-1384, 1984.

This paper presents and analyses a parallizable algorithm for solving Markov chains that arise in queuing models of loosely coupled systems.

(8) David C. Fisher, *In Three-Dimensional Space, the Time Required to Add N Numbers is* $O(N^{1/4})$, TR-1431, August, 1984.

How quickly can the sum of N numbers be computed with sufficiently many processors? The traditional answer is $t = O(\log N)$. However, if the processors are in R^d (usually $d \leq 3$), addition time and processor volume are bounded away from zero, and transmission speed and processor length are bounded, $t \geq O(N^{1/d+1})$.

(9) Dianne P. O'Leary, G. W. Stewart, *On the Determinacy of a Model for Parallel Computation*, TR-1456, November, 1984. (Obsolete: see TR-1553)

In this note we extend a model of Karp and Miller for parallel computation. We show that the model is deterministic, in the sense under different scheduling regimes each process in the computation consumes the same input and generates the same output. Moreover, if the computation halts, the final state is independent of scheduling.

(10) Dianne P. O'Leary, *Systolic Arrays for Matrix Transpose and Other Reorderings*, TR-1481, March, 1985.

In this note, a systolic array is described for computing the transpose of an $n \times n$ matrix in time $3n-1$ using n^2 switching processors and n^2 bit buffers. A one-dimensional implementation is also described. Arrays are also given to take a matrix in by rows and put it out by diagonals, and vice versa.

(11) Dianne P. O'Leary, G. W. Stewart, *Assignment and Scheduling in Parallel Matrix Factorization*, TR-1486, April, 1985.

We consider in this paper the problem of factoring a dense $n \times n$ matrix on a network consisting of P MIMD processors when the network is smaller than the number of elements in the matrix ($P < n^2$). The specific example analyzed is a computational network that arises in computing the LU, QR, or Cholesky factorizations. We prove that if the nodes of the network are evenly distributed among processors and if computations are scheduled by a round-robin or a least-recently-executed scheduling algorithm, then optimal order of speed-up is achieved. However, such speed-up is not necessarily achieved for other scheduling algorithms or if the computation for the nodes is inappropriately split across processors, and we give examples of these phenomena. Lower bounds on execution time for the algorithm are established

(12) Dianne P. O'Leary, G. W. Stewart, *From Determinacy to Systaltic Arrays*, TR-1553, August, 1985.

In this paper we extend a model of Karp and Miller for parallel computation. We show that the extended model is deterministic, in the sense that under different scheduling regimes each process in the computation consumes the same input and generates the same output. Moreover, if the computation halts, the final state is independent of scheduling. The model is applied to the generation of precedence graphs, from which lower time bounds may be deduced, and to the synchronization of systolic arrays by local rather than global control.

(13) David C. Fisher, *Matrix Computation on Processors in One, Two, and Three Dimensions*, TR-1556, August, 1985.

Suppose a problem is to be solved on a d-dimensional parallel processing machine. Assume that transmission speed is finite. Under this and other "real world" assumptions, if a problem requires I inputs, K outputs and T computations, then time required to solve the problem is greater than or equal to $O(\max(I^{1/d}, K^{1/d}, T^{1/(d+1)}))$. Algorithms for certain matrix computations are developed. The problems are divided into ***atoms***. The algorithms are described and analyzed with the use of ***step*** and ***processor assignment functions***. These assign each atom to a step and a processor. Here is a table showing the time for algorithms presented in this paper:

Problem	Linear Grid (1-D)	Square Grid (2-D)	Cubic Grid (3-D)
Summation of k numbers	$O(k)$	$O(k^{1/2})$	$O(k^{1/3})$
Multiply a $k \times k$ matrix by a k vector	$O(k^2)$	$O(k)$	$O(k^{2/3})$
Multiply two $k \times k$ matrices	$O(k^2)$	$O(k)$	$O(k^{3/4})$
Cholesky factorization of a $k \times k$ matrix	——	$O(k)$	——

Except for matrix multiplication in 3-dimensions, these times are a constant multiple of the lower bounds. Programs are given which will execute these algorithms on an appropriate parallel processing machine.

(14) D. P. O'Leary, G. W. Stewart, R. van de Geijn, *DOMINO: A Message Passing Environment for Parallel Computation*. TR-1648, April, 1986.

This report is a description of DOMINO, a system to coordinate computations on a network of processors. It implements an extension of a model of parallel computation (Karp and Miller, SIAM J. Appl. Math., 1966), in which computations are synchronized by messages passed between the processes performing the computations. The system is organized in such a way that meaningful debugging can be done on a single processor. In order to make DOMINO transportable, system dependent features have been isolated in two interfaces.

(15) G. W. Stewart, *Communication in Parallel Algorithms: An Example*, April, 1986.

The purpose of this note is to describe and analyze a parallel algorithm for computing the QR factorization of an $n \times p$ matrix. The algorithm is designed to run on a ring of r processors that communicate by message passing.

(16) Dianne P. O'Leary, Roger Pierson, G. W. Stewart, Mark Weiser, *The Maryland Crab: A Module for Building Parallel Computers*, UMIACS-TR-86-9 and CS-TR-1660, University of Maryland, April, 1986.

There is not yet a consensus on the best way to interconnect a large number of general purpose processors to construct a parallel computer. Both of the standard strategies have major disadvantages: shared memory designs suffer from bus contention; nonshared memory designs have a relatively large communication overhead even for a small number of processors. In this work we present the design for a module, called the Crab, from which machines with *locally* shared memory can be constructed. A Crab consists of a processor, some local non-shared memory, a memory module which can be accessed by its neighboring Crabs, and circuitry to manage shared memory access. Crabs may be connected in arbitrary geometries, e.g., rings, hypercubes, etc. This design greatly reduces the overhead in passing information between neighboring processors and allows scaling to a large number of processors without problems of bus contention. Other features of the design are a global bus, useful for broadcast messages, and interprocessor FIFO interrupt queues.

(17) G. W. Stewart, *A Parallel Implementation of the QR Algorithm*, CS-TR-1662, University of Maryland, May, 1986.

In this paper a parallel implementation of the QR algorithm for the eigenvalues of a non-Hermitian matrix is proposed. The algorithm is designed to run efficiently on a linear array of processors that communicate by accessing their neighbors' memory. A module for building such arrays, the Maryland Crab, is also described.

(18) D. P. O'Leary, *Parallel Implementation of the Block Conjugate Gradient Algorithm*, CS-TR-1676, UMIACS-TR-86-14, University of Maryland, June, 1986.

The conjugate gradient algorithm is well-suited for vector computation but, because of its many synchronization points and relatively short message packets, is more difficult to implement for parallel computation. In this work we introduce a parallel implementation of the block conjugate gradient algorithm. In this algorithm, we carry a block of vectors along at each iteration, reducing the number of iterations and increasing the length of each message. On machines with relatively costly message passing, this algorithm is a significant improvement over the standard conjugate gradient algorithm.

(19) G. W. Stewart, *Bounding Errors in the Solution of a 2×2 System of Equations*, CS-TR-1702, University of Maryland, August, 1986.

This paper solves the small but important problem of assessing the accuracy of the computed solutions of 2×2 systems. The result has applications in areas such as computational geometry, where parallelization is an important issue.

(20) John Michael Conroy *Parallel Direct Solution of Sparse Linear Systems of Equations*, CS-TR-1714, University of Maryland, October, 1986.

We consider the solution of certain sparse linear systems of equations on a parallel computer. These systems arise when using a finite difference or finite element method to solve partial differential equations. We consider both banded matrices and sparse matrices whose graphs are two dimensional meshes. The algorithms given are designed for linear and square arrays of processors.

Two fine grain algorithms for the Cholesky decomposition of band matrices with wide bandwidth are described. We discuss two techniques to perform a Cholesky factorization using a $p \times p$ array of processors, where $p < m$, the semibandwidth of the matrix. The first method uses a folding technique to map a systolic array onto a square array of processors. The second algorithm uses a torus wrap of the band matrix. These methods are also modified to allow for medium grain computation. At most $O(m^{2})$ parallelism is exploited by these algorithms.

Algorithms to yield significant parallelism for narrow banded systems are also derived. The first class of algorithms yield $O(n^{1/2}/m^{1/2})$ parallelism using a linear array of processors, where n is the size of the matrix. Two variations are considered. They have similar complexity, but have different numerical properties, and a backwards error analysis is given. The second method, a fine grain algorithm, allows for up to $O(n^{1/2}m^{3/2}$ parallelism using a rectangular array. This method has been implemented on the Zmob at the University of Maryland. Empirical results of this implementation are discussed.

Finally, an efficient medium grain algorithm for the method of nested dissection for an $N \times N$ mesh is given. The algorithm exploits the ability of a processor to perform communication and computation simultaneously. The strength of the method is its ability to pipeline much of the computation. This method has an asymptotic efficiency of about 49% using a square array of P^2 processors, where $P << N$.

II. Presentations

(1) D. P. O'Leary, *Solving Mesh Problems on Parallel Computers*,
Bell Laboratory, Murray Hill, N.J., January, 1983
IBM T. J. Watson Laboratory, Yorktown Heights, N.Y., January, 1983.

(2) G. W. Stewart, *A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix* (invited), Symposium on Numerical Analysis and Computational Complex Analysis, Zurich, Switzerland, August, 1983. Also presented at North Carolina State University, September, 1983, and at University of Houston, November, 1983,

(3) G. W. Stewart, *The Structure of Nearly Uncoupled Markov Chains* (invited), International Workshop on Systems Modeling, Pisa, Italy, September, 1983.

(4) G. W. Stewart, *Data Flow Algorithms for Parallel Matrix Computations* (invited), SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, November, 1983.

(5) D. P. O'Leary, *Parallel Computations for Sparse Linear Systems* (minisymposium invitation), SIAM 1983 Fall Meeting, Norfolk, VA, November, 1983.

(6) D. C. Fisher, *Numerical Computations on Multiprocessors with Only Local Communications* (poster session), SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, November, 1983.

(7) G. W. Stewart, *Parallel Computations on the ZMOB*, Annual meeting of CER participants, University of Utah, March, 1984.

(8) D. P. O'Leary, *Data-flow Algorithms for Matrix Computations* (minisymposium invitation), ACM SIGNUM Conference on Numerical Computations and Mathematical Software for Microcomputers, Boulder, Colorado, March, 1984.

(9) D. P. O'Leary, *Solution of Matrix Problems on Parallel Computers* (invited presentation), Gatlinburg IX Meeting on Numerical Linear Algebra, Waterloo, Ontario, Canada, July, 1984. Also presented at Oak Ridge National Laboratory, September, 1984; National Bureau of Standards, Boulder, Colorado, March, 1984; Yale University, November, 1984; Cornell University, January, 1985; Courant Institute, February, 1985.

(10) G. W. Stewart, *The Data-Flow Approach to Matrix Computations*, Los Alamos Scientific Laboratory, October, 1984.

(11) G. W. Stewart, *The Impact of Computer Architecture on Statistical Computing*, (invited) SIAM/ISA/ASA Conference on Frontiers of Statistical Computing, October, 1984.

(12) G. W. Stewart, *Determinacy*, (invited) Symposium in Honor of G. Dahlquist, Stockholm, January, 1985.

(13) D. C. Fisher, *Fast Matrix Multiplication on Square and Cubic Grids of Processors*, SIAM Conference on Applied Linear Algebra, Raleigh, April, 1985.

(14) G. W. Stewart, *The Parallel Solution of Sparse Unsymmetric Eigenvalue Problems*, (invited) IBM Workshop on Large Eigenvalue Problems, Oberlech, Austria, July, 1985.

(15) D. P. O'Leary, *A Testbed for Parallel Algorithm Development*, (invited) SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, November, 1985.

(16) D. P. O'Leary, G. W. Stewart, Robert van de Geijn, *Domino, a Transportable System for Parallel Computing*, Army Research Conference on Parallel Computing and Medium Scale Multiprocessors, Stanford, CA, January, 1986.

(17) D. P. O'Leary, *Parallel Computation and Linear Programming*, Workshop on Future Directions in Mathematical Programming, Naval Postgraduate School, Monterey, California, February, 1986.

(18) G. W. Stewart, *Communication in Parallel Algorithms: An Example*, 18th Symposium on the Interface between Computer Science and Statistics, College Station, Colorado, March, 1986.

(19) G. W. Stewart, *Parallel Scientific Computing*, University of Colorado Conference on Computer Science, Boulder, CO, March, 1986.

(20) D. P. O'Leary, *Fine and Medium Grained Parallel Algorithms for Matrix QR Factorization*, Stichting Mathematisch Centrum, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, May, 1986.

(21) G. W. Stewart, *A Parallel Implementation of the QR Algorithm*, International Conference on Vector and Parallel Computing, Loen, Norway, June, 1986.

(22) D. P. O'Leary, *Parallel Implementation of the Block Conjugate Gradient Algorithm*, International Conference on Vector and Parallel Computing, Loen, Norway, June, 1986.

(23) Robert van de Geijn, *Message Passing on the Butterfly*, Butterfly Users Group Meeting, Boston, Massachusetts, October, 1986.

(24) G. W. Stewart, *Domino: a Parallel Message Passing Environment*, Workshop on Software Issues in Parallel Computing, Norfolk, November, 1986.

(25) D. P. O'Leary, *Some Small Problems in Parallel Computing*, University of Illinois, January, 1987.

(26) G. W. Stewart, *Parallel Computing*, Chinese Academy of Science, January, 1987.

(27) D. P. O'Leary, *Some Small Problems in Parallel Computing*, Pennsylvania State University, March, 1987.

(28) D. P. O'Leary, *Domino: A Portable Parallel Programming Environment*, Supercomputing Research Center, March, 1987.

(29) G. W. Stewart, *Communications and Matrix Computations*,
Workshop on Numerical Computation, Northwestern University, March, 1987.
Argonne National Laboratories, March, 1987
University of Illinois, March, 1987

IV. Publications

(1) G. W. Stewart, "Computing the CS Decomposition of a Partitioned Orthonormal Matrix," *Numerische Mathematik* 40 (1982) 297-306.

(2) D. P. O'Leary, "Ordering schemes for parallel processing of certain mesh problems," *SIAM Journal on Scientific and Statistical Computing* 5 (1984) 620-632.

(3) D. P. O'Leary, G. W. Stewart, "Data-flow algorithms for parallel matrix computations," *Communications of the ACM,* 28 (1985) 840-853.

(4) G. W. Stewart, "A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix," *SIAM Journal on Scientific and Statistical Computing,* 6 (1985) 853-864.

(5) D. P. O'Leary, R. E. White, "Multi-splittings of matrices and parallel solution of linear systems," *SIAM Journal on Algebraic and Discrete Methods,* 6 (1985) 630-640.

(6) G. W. Stewart "A Note on Complex Division," *ACM Transactions on Mathematical Software* 11 (1985) 238-241.

(7) D. P. O'Leary and G. W. Stewart, "Assignment and Scheduling in Parallel Matrix Factorization," *Linear Algebra and Its Applications,* 77 (1986) 275-300.

(8) G. W. Stewart, "Communication in Parallel Algorithms: An Example," *Proceedings of the 18th Symposium on the Interface between Computer Science and Statistics,* American Statistical Association, 1986, 11-14.

(9) D. P. O'Leary, R. van de Geijn, G. W. Stewart, "DOMINO: A Transportable Operating System," Proceedings of the Army Conference on Parallel Processing, Stanford, CA, January, 1986, A. Wouk, ed., to appear.

(10) D. P. O'Leary, "Systolic Arrays for Matrix Transpose and Other Reorderings," *IEEE Transactions on Computers,* C-36 (1987) 117-122.

(11) G. W. Stewart, "A Parallel Implementation of the QR Algorithm," *Parallel Computing,* to appear.

(12) D. P. O'Leary, "Parallel Implementation of the Block Conjugate Gradient Algorithm," *Parallel Computing,* to appear.

(13) D. P. O'Leary and G. W. Stewart, "From Determinacy to Systaltic Arrays," *IEEE Transactions on Computers,* to appear.